

# トロン技術者認定試験問題

ucodeQR



00030168

## A. 注意事項

1. **試験開始の合図があるまで、この問題冊子を開いてはいけません。**
2. 試験開始と終了の時刻は試験監督員の時計が基準となります。試験監督員の指示に従いなさい。
3. 本冊子の本文は32ページです。試験中にページの落丁、乱丁、印刷不鮮明、および解答用紙の汚れ等に気づいた場合は手を挙げて試験監督員に知らせなさい。
4. **問題に関する質問には答えることはできません。** 文意どおりに解釈して解答しなさい。
5. **「受験票」は、必ず机の上に置きなさい。**
6. 「受験票」のほかに試験時間中、机の上に置けるものは、「B または HB の黒鉛筆または黒色シャープペンシル」「プラスチック製の消しゴム」「鉛筆削り（電動式を除く）」「時計」「眼鏡」です。これ以外の所持品を置いてはいけません。
7. 解答用紙には氏名欄があるので、試験監督員の指示に従って、正しく記入しなさい。
8. 解答用紙の試験会場欄、受験日欄が正しいことを、試験監督員の指示に従って確認しなさい。
9. 解答用紙の受験番号欄に、自分の受験番号が記載されており、かつ、正しくマークされていることを確認しなさい。
10. 13時30分以降は、試験終了後の確認作業が終了するまで退室を認めません。**試験中の発病又はトイレ等やむを得ない場合には、手を挙げて試験監督員の指示に従いなさい。**ただし、一時退室が認められた場合でも、試験教室以外での受験はできません。試験時間の延長も認められません。
11. 試験時間中は、試験監督員の指示に従いなさい。従わない場合は退室させることがあります。

## B. 本冊子、解答用紙について

1. 本冊子は、トロン技術者認定試験のためのものです。
2. 本冊子には、計25問の問題が収録されています。**すべての問題を解答しなさい。**
3. 解答用紙は本冊子とは別に1枚あり、マークシート方式による記入となります。
4. 本冊子の余白等は適宜利用してよいですが、どのページも切り離してはいけません。
5. **本冊子、解答用紙は持ち帰ることはできません。**
6. 以下の欄に受験番号と氏名を記入しなさい。

受験番号	
氏名	

解答上の注意が裏表紙に記載してあるので、この問題冊子を**左側から裏返**して必ず読みなさい。ただし、問題冊子を開いてはいけません。



## 注意事項

- ・ 特に記載されていない限り、 $\mu$ ITRON とは  $\mu$ ITRON4.0 と考えなさい。
- ・ 特に記載されていない限り、「タスク」とはプログラムの並行実行の単位と考えなさい。
- ・ 特に記載されていない限り、「システムコール」と「サービスコール」は同じ意味と考えなさい。
- ・ 特に記載されていない限り、マルチプロセッサでの動作は考慮せずに解答しなさい。
- ・ プログラムリストに含まれる /\* 略 \*/ の箇所は、何らかの処理が省略されているものとして解答しなさい。

出題区分 1 (20問 各3点)

問01 ~ 問20

出題区分 2 (5問 各8点)

問21 ~ 問25

## 問 0 1

T-Kernel ( $\mu$ ITRON) のタスクスケジューリングに関する以下の説明のうち、誤っているものをすべて選べ。

### 選択肢

1. T-Kernel ( $\mu$ ITRON) では、同じ優先度を持つタスク間では、先に実行できる状態 (実行状態または実行可能状態) になったタスクの方が高い優先順位を持つ。これを FCFS (First Come First Served) 方式という。
2. T-Kernel ( $\mu$ ITRON) では、優先度の高いタスクが実行される優先度方式でタスクスケジューリングを行う。ただし、同じ優先度のタスクが複数ある場合はラウンドロビン方式でスケジューリングされる。
3. T-Kernel ( $\mu$ ITRON) では、優先度ベースのスケジューリングを行っている。実行されるのは、その時点で最も優先順位の高いタスクである。何らかの要因で実行できなくなった場合に、次に優先順位の高いタスクに実行権が渡される。この実行権の切り替えをディスパッチという。
4. 現在実行中のタスクが資源要求をし、資源が獲得できず待ち状態に入った場合などには、優先度の低いタスクに実行権が渡されることもある。このように、実行中のタスクが何らかの理由で実行権を他のタスクにわたすことをプリエンプトという。

## 問 0 2

T-Kernel ( $\mu$ ITRON) を使用したシステムにおいて、タスク A、タスク B、タスク C、タスク D の 4 つのタスクがあり、いずれも同じ ID 番号のイベントフラグの待ち解除条件の成立を待っているものとする。

今、この ID 番号のイベントフラグを対象に tk\_set\_flg (set\_flg) サービスコールを発行した。

この tk\_set\_flg (set\_flg) サービスコールの発行により、イベントフラグの待ちが解除されるタスクを、選択肢の中からすべて選べ。

tk\_set\_flg (set\_flg) によってセットしたビットパターン：  
0xF000F000

イベントフラグの状態：  
ビットパターン： 0x00000000  
イベントフラグ属性： TA\_TPRI | TA\_WMUL

タスク A の状態：  
待ちビットパターン： 0x00000001  
優先度： 3  
待ちモード： TWF\_ORW | TWF\_CLR

タスク B の状態：  
待ちビットパターン： 0xF0F0F0F0  
優先度： 5  
待ちモード： TWF\_ANDW | TWF\_CLR

タスク C の状態：  
待ちビットパターン： 0xFFFF0000  
優先度： 6  
待ちモード： TWF\_ORW | TWF\_CLR

タスク D の状態：  
待ちビットパターン： 0xF0000000  
優先度： 4  
待ちモード： TWF\_ANDW | TWF\_CLR

※T-Kernel と  $\mu$ ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel の tk\_wai\_flg のパラメータとして指定する待ちモード TWF\_CLR は、 $\mu$ ITRON のイベントフラグ属性として指定する TA\_CLR と同じ機能である。

選択肢

1. タスク A
2. タスク B
3. タスク C
4. タスク D

## 問 0 3

T-Kernel ( $\mu$ ITRON)のメールボックスの説明として正しいものを、以下の中から一つ選べ。

選択肢

1. メールボックスは、メールボックスごとに用意されたバッファに対するメッセージ本体の書き込みと、読み出しを通して同期と通信を行うためのオブジェクトである。
2. メールボックスでは、送信側タスクは、受信側タスクのタスク ID を指定して、メッセージを送信する。
3. メッセージを受信するタスクがメールボックスの待ち行列に並ぶ際の並び方は、タスクの優先度順で固定されている。
4. メールボックスを削除する場合、対象メールボックスの中にメッセージが残っていても、メールボックスを削除することができる。

## 問 0 4

T-Kernel で動作するプログラムにおいて、優先度 10 のタスク B が実行可能状態となり、その 2 ミリ秒後に優先度 10 のタスク C、更にその 3 ミリ秒後に優先度 5 のタスク A が実行可能状態となった。

タスク B が実行可能状態となった時刻から、タスク C が再び待ち状態になるまでの時間として正しいものを、選択肢の中から一つ選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ タスク A は 25 ミリ秒の間処理を行った後、再び待ち状態に入る。
- ・ タスク B は 20 ミリ秒の間処理を行った後、再び待ち状態に入る。
- ・ タスク C は 10 ミリ秒の間処理を行った後、再び待ち状態に入る。
- ・ すべてのタスクは、タスクスライスタイムが 10 ミリ秒に設定されている。

※T-Kernel と  $\mu$ ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel のタスクスライスタイムの仕様は以下の通りである。

スライスタイムはタスクのラウンドロビンスケジューリングのための機能である。タスクが slicetime (スライスタイムとして指定された時間) 以上の時間、連続して実行されると、同じ優先度を持つタスクの中で最低の優先順位となり、自動的に実行権を次のタスクに譲る。

より優先度の高いタスクによって実行権が奪われている間は連続実行時間としてカウントされない。また、より高い優先度のタスクによって実行権が奪われても、不連続と扱わない。つまり、より高い優先度のタスクによって実行権が奪われている間は無視して、実行時間をカウントする。

選択肢

1. 40 ミリ秒
2. 45 ミリ秒
3. 47 ミリ秒
4. 50 ミリ秒

## 問 0 5

T-Kernel ( $\mu$ ITRON)のタスクXとタスクYが、資源Aと資源Bという2つの資源を共有しているため、セマフォを利用して排他制御を行いたい。

タスクX、タスクYが、それぞれ選択肢にあるような順番で tk\_wai\_sem、tk\_sig\_sem を発行する場合、デッドロックが発生する可能性がないものはどれか。選択肢の中から一つ選べ。

ただし、獲得、解放する資源数はすべて1であるとし、タイムアウトなどの引数は省略してある。

※T-Kernel と  $\mu$ ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ サービスコールの名称は tk\_wai\_sem を wai\_sem のように読み替える。

選択肢

1. タスクX : tk\_wai\_sem(A) → tk\_wai\_sem(B) → 資源 A, B の利用 → tk\_sig\_sem(B) → tk\_sig\_sem(A)  
タスクY : tk\_wai\_sem(B) → tk\_wai\_sem(A) → 資源 A, B の利用 → tk\_sig\_sem(A) → tk\_sig\_sem(B)
2. タスクX : tk\_wai\_sem(A) → tk\_wai\_sem(B) → 資源 A, B の利用 → tk\_sig\_sem(A) → tk\_sig\_sem(B)  
タスクY : tk\_wai\_sem(B) → tk\_wai\_sem(A) → 資源 A, B の利用 → tk\_sig\_sem(B) → tk\_sig\_sem(A)
3. タスクX : tk\_wai\_sem(A) → tk\_wai\_sem(B) → 資源 A, B の利用 → tk\_sig\_sem(B) → tk\_sig\_sem(A)  
タスクY : tk\_wai\_sem(B) → tk\_wai\_sem(A) → 資源 A, B の利用 → tk\_sig\_sem(B) → tk\_sig\_sem(A)
4. タスクX : tk\_wai\_sem(A) → tk\_wai\_sem(B) → 資源 A, B の利用 → tk\_sig\_sem(B) → tk\_sig\_sem(A)  
タスクY : tk\_wai\_sem(A) → tk\_wai\_sem(B) → 資源 A, B の利用 → tk\_sig\_sem(B) → tk\_sig\_sem(A)

## 問 0 6

T-Kernel の時間管理機能に関する以下の説明のうち、誤っているものを一つ選べ。

### 選択肢

1. システム動作中に tk\_set\_tim を使ってシステム時刻を更新した場合、タイムアウトのパラメータで指定された相対時間は変化しない。たとえば、60 秒後にタイムアウトするように指定した場合、タイムアウト待ちの間に tk\_set\_tim で時間を 60 秒進めてもそこでタイムアウトすることはなく、タイムアウトの指定から 60 秒後にタイムアウトする。
2. tk\_get\_otm はシステム稼働時間を取得する。システム稼働時間は、システム起動時からの単純増加する時間であり、tk\_set\_tim による時刻設定に影響されない。
3. tk\_get\_tim ではシステム時刻の現在の値を読み出す。システム時刻は、1985 年 1 月 1 日 0 時 0 分 0 秒 (GMT) からの通算の秒数である。
4. tk\_set\_tim、tk\_get\_tim、および tk\_get\_otm システムコールでは、時刻の指定には以下のよう  
に定義されたシステム時刻型 (SYSTIM 型) が用いられる。

```
typedef struct systim {  
    W      hi;  
    UW     lo;  
} SYSTIM;
```

## 問 07

T-Kernel ( $\mu$ ITRON)のタスクの削除と終了に関する以下の説明のうち、正しいものをすべて選べ。

選択肢

1. 実行中のタスクが自タスクを終了する場合は、tk\_ext\_tsk (ext\_tsk) を使用する。
2. 実行中のタスクが自タスクを削除するシステムコールはない。
3. 実行中のタスクが別のタスクを終了させる場合は、tk\_del\_tsk (del\_tsk) を使用する。
4. タスクを削除するシステムコールが発行されても、未登録状態にならずに休止状態になる場合がある。

## 問 0 8

T-Kernel ( $\mu$ ITRON)のアラームハンドラの処理に関する以下の説明のうち、正しいものを一つ選べ。

### 選択肢

1. アラームハンドラは生成した時点では起動時刻が設定されておらず、動作は停止している。アラームハンドラを動作させるためには tk\_sta\_alm (sta\_alm) によって起動時刻を設定しなければならない。
2. アラームハンドラの実行が終了すると、そのアラームハンドラは自動的に削除される。
3. アラームハンドラの中でシステムコールを発行することにより、それまで実行状態であったタスクがその他の状態に移行し、代わりに別のタスクが実行状態となった場合は、即座にタスクディスパッチが行われる。
4. アラームハンドラは、アラームハンドラを生成したタスクのタスク部として実行され、アラームハンドラの中でも、待ち状態に入るシステムコールを発行することができる。

## 問 0 9

T-Kernel ( $\mu$ ITRON)の以下のシステムコールのうち、ディスパッチを起こす可能性があるものを、以下の中からすべて選べ。

選択肢

1. tk\_set\_flg (set\_flg)
2. tk\_clr\_flg (clr\_flg)
3. tk\_wai\_flg (wai\_flg)
4. tk\_del\_flg (del\_flg)

## 問 10

T-Kernel ( $\mu$ ITRON)のメモリプール管理機能に関する以下の説明のうち、正しいものをすべて選べ。

選択肢

1. 可変長メモリプール領域に空きがない状態で、可変長メモリプールからメモリブロックを獲得しようとしたタスクは、メモリブロックサイズとして指定した連続したメモリブロックが獲得できるようになるまで可変長メモリブロックの獲得待ち状態となる。
2. 可変長メモリプールでメモリブロックの獲得を待っているタスクは、要求ブロックサイズの小さい順にメモリブロックを獲得する。これによりメモリの効率的な利用を実現している。
3. 固定長メモリプール領域に空きがない状態で、固定長メモリプールからメモリブロックを獲得しようとしたタスクは、獲得メモリブロック数として指定したメモリブロックが獲得できるようになるまで固定長メモリブロックの獲得待ち状態となる。
4. メモリ獲得を待っているタスクが存在するメモリプールであっても削除することができる。この場合、メモリ獲得待ち状態にあったタスクにはエラー(E\_DLT)が返される。

## 問 1 1

T-Kernel ( $\mu$ ITRON)のセマフォ機能に関する以下の説明のうち、誤っているものをすべて選べ。

選択肢

1. 割込みハンドラから `tk_sig_sem (sig_sem)` を呼び出すことによって、割込みが発生したことをタスクに通知することができる。
2. あるタスクにおいて複数のセマフォから資源を獲得していた場合、一回の `tk_sig_sem (sig_sem)` の呼び出しで複数のセマフォに資源を返却することができる。
3. 優先度の異なる複数のタスク間で1つのセマフォ資源を共有している場合、優先度逆転が発生することがある。
4. `tk_wai_sem (twai_sem)` を使用し、タイムアウト処理を適切に行うことによってこのセマフォに関して発生するデッドロックを防ぐことができる。

## 問 1 2

T-Kernel ( $\mu$ ITRON)のtk\_sus\_tsk (sus\_tsk) を発行することによる対象タスクの状態遷移として正しいものを、選択肢の中からすべて選べ。

なお、選択肢の表記は次のとおりである。

[tk\_sus\_tsk (sus\_tsk) 発行前のタスクの状態] → [tk\_sus\_tsk (sus\_tsk) 発行後のタスクの状態]

選択肢

1. 待ち状態 → 二重待ち状態
2. 実行可能状態 → 強制待ち状態
3. 二重待ち状態 → 強制待ち状態
4. 強制待ち状態 → 強制待ち状態

## 問 1 3

T-Kernel ( $\mu$  ITRON)のランデブポートでは、tk\_fwd\_por (fwd\_por) によってランデブを別のランデブポートに回送することができる。

T-Kernel ( $\mu$  ITRON)のランデブポートに関する以下の説明のうち、誤っているものを一つ選べ。

選択肢

1. ランデブを回送する場合、ランデブ成立のための呼び出し側条件は tk\_fwd\_por (fwd\_por) の呼び出しの時に指定したビットパターンとなる。
2. 回送先のランデブポートは、回送前のランデブに使っていたランデブポートと同じランデブポートであっても構わない。
3. 回送先のランデブポートの返答メッセージの最大サイズが、回送前のランデブに使っていたランデブポートの返答メッセージの最大サイズを超えていた場合、tk\_fwd\_por (fwd\_por) にはエラーが返される。
4. 回送先のランデブポートに受付待ちタスクが無かった場合や、受付待ちタスクがあってもランデブ成立条件が満たされなかった場合には、tk\_fwd\_por (fwd\_por) 発行タスクは待ち状態となる。

## 問 1 4

T-Kernel ( $\mu$ ITRON) のイベントフラグ機能に関する以下の説明のうち、誤っているものを一つ選べ。

選択肢

1. イベントフラグの待ち行列は、FIFO 順、タスクの優先度順をイベントフラグ生成時に設定することができる。生成後に待ち行列の FIFO 順、タスクの優先度順を変更することはできない。
2. イベントフラグでは、一つのイベントフラグで同時に複数のタスクが待ち状態になることができる。ただし、イベントフラグ属性の指定により、複数のタスクが同時に待ち状態になることを禁止することもできる。
3. `tk_set_flg (set_flg)` においてセットするビットパターンを全ビット 0 とした場合や、`tk_clr_flg (clr_flg)` においてクリアするビットパターンを全ビット 1 とした場合には、対象イベントフラグに対して何の操作も行わないため、エラーとなる。
4. イベントフラグがすでに待ち解除条件を満たしている場合、`tk_wai_flg (wai_flg)` を発行したタスクは待ち状態にならずに実行を続ける。

## 問 1 5

T-Kernel ( $\mu$ ITRON)の周期ハンドラ機能に関する以下の説明のうち、正しいものをすべて選べ。

選択肢

1. tk\_sta\_cyc (sta\_cyc) を呼び出さなくても、周期ハンドラを起動することができる。
2. 周期ハンドラの起動周期は、周期ハンドラを起動した時刻を基準として、次に周期ハンドラを起動する時刻を指定するための相対時間と解釈する。そのため、周期ハンドラが起動される時刻の間隔は起動周期以上になる。
3. 周期ハンドラ属性に TA\_PHS が設定されている場合は、一旦周期ハンドラを停止した後に再開しても、生成時に決められた時刻ごとに周期ハンドラの処理が実行される。
4. 周期ハンドラで呼び出したシステムコールによってディスパッチ要求があった場合、周期ハンドラの処理が終了した後にディスパッチされる。

## 問 1 6

T-Kernel ( $\mu$ ITRON) のミューテックス機能に関する以下の説明のうち、誤っているものをすべて選べ。

選択肢

1. ミューテックスは、排他制御に伴う上限のない優先度逆転を防ぐための機構として、優先度継承プロトコルと、優先度上限プロトコルをサポートする。
2. ミューテックスの待ちタスクのキューイングには、FIFO またはタスク優先度順を指定できる。
3. ミューテックスの操作に伴ってタスクの現在優先度が変更された結果、優先度を変更されたタスクのタスク優先度順の待ち行列の中での順序が変化した場合でも、優先度を変更されたタスクの待ち解除は発生しない。
4. 対象ミューテックスにおいてロック待ちをしているタスクがある場合、tk\_del\_mtx (del\_mtx) によってミューテックスを削除することはできない。

## 問 1 7

μITRON のタスク例外処理機能に関する説明として正しいものを、以下の中からすべて選べ。

選択肢

1. タスク例外処理ルーチンはタスクごとに一つだけ登録することができる。
2. 待ち状態のタスクにタスク例外処理を要求した場合は、タスク例外処理ルーチンだけ即座に処理を実行する。
3. タスク例外処理ルーチンは、GPU ロック状態で起動される場合がある。
4. タスク例外処理ルーチンは、ディスパッチ禁止状態で起動される場合がある。

## 問 1 8

T-Kernel ( $\mu$ ITRON)の tk\_dly\_tsk (dly\_tsk) に関する以下の説明のうち、正しいものを一つ選べ。  
ただし、パラメータの基準時間は1ミリ秒とする。

選択肢

1. tk\_dly\_tsk (dly\_tsk) を発行した場合、指定された時間経過すると戻値が E\_TMOUT で終了する。
2. パラメータに 100 を指定して tk\_dly\_tsk (dly\_tsk) を発行した場合、自タスクは 100 ミリ秒以上待ち状態になる。
3. パラメータに TMO\_FEVR を指定して tk\_dly\_tsk (dly\_tsk) を発行した場合、他のタスクなどから待ちを解除されない限り永久に待ち状態になる。
4. tk\_dly\_tsk (dly\_tsk) を発行したタスクが強制待ち状態にされた場合、時間経過のカウントも停止される。

## 問 1 9

T-Kernel ( $\mu$ ITRON)のハンドラ、ディスパッチャ、タスクの各処理について、優先度の高い順に並んでいるものを以下の中から一つ選べ。

選択肢

1. ディスパッチャ、割込みハンドラ、タスク
2. ディスパッチャ、タスク、タイムイベントハンドラ
3. 割込みハンドラ、ディスパッチャ、タスク
4. CPU 例外ハンドラ、タスク、ディスパッチャ

## 問 2 0

T-Kernel Standard Extension のプロセスのメインタスクやサブタスクの優先度として、タスク生成時に 0~255 の優先度が与えられる。このうち、「128~191」の優先度を与えられたタスクのスケジューリング方式の説明として正しいものを、以下の中から一つ選べ。

### 選択肢

1. このグループ内全体でラウンドロビンスケジューリングが行われ、すべてのタスクは平等にスケジューリングされる。
2. このグループ内全体でラウンドロビンスケジューリングが行われ、優先度は相対的なスケジューリングの頻度を示す。
3. 優先度順に厳密にスケジューリングされ、同一優先度の場合は FCFS (First Come First Served) 方式でスケジューリングされる。
4. 優先度順に厳密にスケジューリングされ、同一優先度の場合はラウンドロビンで平等にスケジューリングされる。

## 問 2 1

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。

このとき、下記の設問 1、設問 2 に解答せよ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ 可変長メモリプールは正常に生成され、ID が `mplid` に設定されている。
- ・ `mplid` で示される可変長メモリプールの属性は `TA_TFIFO` | `TA_RNGO` である。
- ・ `mplid` で示される可変長メモリプールのメモリプールサイズは 1080 バイトである。
- ・ `taskA` の `tk_get_mpl` は正常に終了し、メモリは獲得される。
- ・ タスクの優先度は全て同じであるとする。
- ・ タスクの優先順位が `taskA`、`taskB`、`taskC`、`taskD` の順になっている状態から実行を開始する。

※T-Kernel と  $\mu$ ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel では、タイムアウトなどの基準時間は 1 ミリ秒を単位としている。
- ・ T-Kernel の `tk_get_mpl` は  $\mu$ ITRON の `tget_mpl` に相当する。
- ・ サービスコールの名称は `tk_dly_tsk` を `dly_tsk` のように読み替える。

### 【設問1】

[ ア ] の処理を実行することによって待ち状態が解除されるタスクを、以下の中から一つ選べ。

選択肢

1. [ ア ] の処理を実行することによって待ち状態が解除されるタスクは存在しない。
2. `taskB`
3. `taskC`
4. `taskD`

### 【設問2】

3番目に `tk_ext_tsk()` が実行されるタスクを、以下の中から一つ選べ。

選択肢

1. `taskA`
2. `taskB`
3. `taskC`
4. `taskD`

```

EXPORT ID      mplid;                /* 可変長メモリプール ID */

void taskA( INT stacd, VP exinf )
{
    UB      *mblk8 = NULL;
    UB      *mblk2 = NULL;

    tk_get_mpl( mplid, 800, &mblk8, TMO_FEVR );
    tk_get_mpl( mplid, 200, &mblk2, TMO_FEVR );

    tk_dly_tsk( 100 );
    tk_rel_mpl( mplid, mblk2 );
    tk_dly_tsk( 200 );
    tk_rel_mpl( mplid, mblk8 );
}

void taskB( INT stacd, VP exinf )
{
    UB      *mblk = NULL;

    tk_get_mpl( mplid, 400, &mblk, 200 );
    tk_dly_tsk( 1000 );
    tk_rel_mpl( mplid, mblk );
    tk_ext_tsk();
}

void taskC( INT stacd, VP exinf )
{
    UB      *mblk = NULL;

    tk_get_mpl( mplid, 100, &mblk, TMO_FEVR );
    tk_dly_tsk( 1000 );
    tk_rel_mpl( mplid, mblk );
    tk_ext_tsk();
}

void taskD( INT stacd, VP exinf )
{
    UB      *mblk = NULL;

    tk_get_mpl( mplid, 200, &mblk, TMO_FEVR );
    tk_dly_tsk( 1000 );
    tk_rel_mpl( mplid, mblk );
    tk_ext_tsk();
}

```

←[ ア ]

## 問 2 2

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。

このとき、プログラム中の [ ア ] を通過してから、[ イ ] に到達するまでにかかる処理時間を、選択肢の中から一つ選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ /\* 略 \*/ の部分の処理にかかる時間は十分に小さく、無視できるものとする。
- ・ 各システムコールの処理にかかる時間は十分に小さく、無視できるものとする。
- ・ タイムティックの処理にかかる時間は十分に小さく、無視できるものとする。
- ・ 実行開始時の各タスクのタスク ID と優先度などは以下のとおりである。

タスク	タスク ID	優先度	タスク状態
taskA	tskidA	1	実行状態
taskB	tskidB	4	休止状態
taskC	tskidC	6	休止状態
taskD	tskidD	8	休止状態

- ・ アラームハンドラとアラームハンドラ ID の対応は以下のとおりである。

アラームハンドラ	アラームハンドラ ID
alm1	almid1
alm2	almid2

※T-Kernel と  $\mu$  ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel では、タイムアウトなどの基準時間は 1 ミリ秒を単位としている。
- ・ T-Kernel の tk\_slp\_tsk は  $\mu$  ITRON の tslp\_tsk に相当する。
- ・ T-Kernel の tk\_sta\_tsk は  $\mu$  ITRON の sta\_tsk、ista\_tsk に相当する。
- ・ T-Kernel の tk\_sta\_tsk の第 2 引数は、stacd としてタスクに渡される。
- ・ T-Kernel の tk\_wup\_tsk は  $\mu$  ITRON の wup\_tsk、iwup\_tsk に相当する。
- ・ サービスコールの名称は tk\_dly\_tsk を dly\_tsk のように読み替える。

選択肢

1. 1500 ミリ秒
2. 1600 ミリ秒
3. 1700 ミリ秒
4. 1800 ミリ秒

```

void    taskA( INT stacd, VP exinf )
{
    /* 略 */                               ← [ ア ]
    tk_dly_tsk( 100 );
    tk_sta_alm( almid1, 800 );
    tk_ext_tsk();
}

void    taskB( INT stacd, VP exinf )
{
    tk_sta_alm( almid2, 800 );
    tk_dly_tsk( stacd );
    tk_ext_tsk();
}

void    taskC( INT stacd, VP exinf )
{
    tk_stp_alm( almid2 );
    tk_sta_alm( almid2, 300 );
    tk_dly_tsk( stacd );
    tk_ext_tsk();
}

void    taskD( INT stacd, VP exinf )
{
    tk_dly_tsk( stacd );
    tk_slp_tsk( stacd );
    /* 略 */                               ← [ イ ]
    tk_ext_tsk();
}

void    alm1( VP exinf )
{
    tk_sta_tsk( tskidB, 200 );
    tk_sta_tsk( tskidC, 400 );
    tk_sta_tsk( tskidD, 600 );
}

void    alm2( VP exinf )
{
    tk_wup_tsk( tskidD );
}

```

## 問 2 3

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。  
プログラムを実行してから 8 秒後の result の数値を解答欄にマークしなさい。  
ただし、プログラムは以下の条件で動作するものとする。

- ・ 起動時には task\_init() が優先度 5 のタスクとして実行される。
- ・ task\_init() 中のすべてのシステムコールは正常に終了する。
- ・ T\_CTSK 構造体のメンバは次の通りとし、これ以外のメンバの記載は省略している。

```
typedef struct t_ctsk {
    VP    exinf;        /* 拡張情報 */
    ATR   tskatr;       /* タスク属性 */
    FP    task;        /* タスク起動アドレス */
    PRI   itskpri;     /* タスク起動時優先度 */
    /* これ以下のメンバの記載は省略 */
} T_CTSK;
```

- ・ T\_CPOR 構造体のメンバは次の通りとし、これ以外のメンバの記載は省略している。

```
typedef struct t_cpor {
    VP    exinf;        /* 拡張情報 */
    ATR   poratr;       /* ランデブポート属性 */
    INT   maxcmsz;     /* 呼出メッセージの最大長(バイト) */
    INT   maxrmsz;     /* 返答メッセージの最大長(バイト) */
    /* これ以下のメンバの記載は省略 */
} T_CPOR;
```

- ・ tk\_dly\_tsk による遅延時間は十分に正確なものとし、遅延時間の不正確さにより解答に影響を与えることはないものとする。
- ・ tk\_dly\_tsk 以外のシステムコールの実行時間やその他のプログラムの実行時間は十分に短いものとし、これらの時間が解答に影響を与えることはないものとする。

※ T-Kernel と  $\mu$  ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel の tk\_cre\_por は  $\mu$  ITRON の acre\_por に相当する。
- ・ T-Kernel の tk\_cre\_tsk は  $\mu$  ITRON の acre\_tsk に相当する。
- ・ T-Kernel の tk\_acp\_por は  $\mu$  ITRON の tacp\_por に相当する。
- ・ T-Kernel の tk\_cal\_por は  $\mu$  ITRON の tcal\_por に相当する。
- ・ T-Kernel では tk\_dly\_tsk の引数をミリ秒単位で指定する。
- ・ サービスコールの名称は tk\_ext\_tsk を ext\_tsk のように読み替える。

```

#define MAX_CAL      8
#define MAX_ACP      3
ID      tskid_cal[MAX_CAL + 1];
ID      tskid_acp[MAX_ACP];
ID      port_id;
INT      result;

void      task_init(INT stacd, VP exinf)      /* 初期化タスク(優先度 5) */
{
    T_CTSK      ctsk_acp = {NULL, TA_HLNG, task_acp, 81, /* 他のメンバは省略 */ };
    T_CTSK      ctsk_cal = {NULL, TA_HLNG, task_cal, 82, /* 他のメンバは省略 */ };
    T_CPOR      cpor = {NULL, TA_TFIFO, 16, 16};
    INT      i;

    result = 9;
    port_id = tk_cre_por(&cpor);

    for(i = 0; i < MAX_ACP; i++) {
        tskid_acp[i] = tk_cre_tsk(&ctsk_acp);
        tk_sta_tsk(tskid_acp[i], 0);
    }
    for(i = 1; i <= MAX_CAL; i++){
        tskid_cal[i] = tk_cre_tsk(&ctsk_cal);
        tk_sta_tsk(tskid_cal[i], i);
    }
    tk_exd_tsk();
}

void      task_acp(INT stacd, VP exinf)
{
    B      msg[4];
    RNO      rdvno;

    for(;;){
        tk_acp_por(port_id, 0x01, &rdvno, msg, TMO_FEVR);
        tk_dly_tsk(msg[0] * 1000);
        tk_rpl_rdv(rdvno, msg, 1);
        tk_dly_tsk(1000);
    }
}

void      task_cal(INT stacd, VP exinf)
{
    B      msg[4];

    tk_dly_tsk(500);
    for(;;){
        msg[0] = stacd;
        tk_cal_por(port_id, 0x01, msg, 1, TMO_FEVR);
        result = stacd;
        tk_dly_tsk(1000);
    }
}

```

## 問 2 4

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。

このとき、プログラム中の [ ア ] を通過してから、タスク C の処理が 2 回終了するまでにかかる時間にもっとも近い値を、選択肢の中から一つ選べ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ taskS は優先度 1 で生成、起動されている。
- ・ taskA、taskB、taskC において省略されている処理の間は、システムコールは発行されていない。
- ・ 待ちを発生する以外のシステムコールの処理にかかる時間は十分に小さく、無視できるものとする。
- ・ タイムティックの処理にかかる時間は十分に小さく、無視できるものとする。

※T-Kernel と  $\mu$ ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel では、タイムアウトなどの基準時間は 1 ミリ秒を単位としている。
- ・ T-Kernel の `tk_slp_tsk(TMO_FEVR)` は  $\mu$ ITRON の `slp_tsk()` に相当する。
- ・ T-Kernel の `tk_cre_tsk` は  $\mu$ ITRON の `acre_tsk` に相当する。
- ・ T-Kernel の `tk_sta_tsk` は  $\mu$ ITRON の `sta_tsk` に相当する。
- ・ T-Kernel の `tk_wup_tsk` は  $\mu$ ITRON の `wup_tsk` に相当する。

選択肢

1. 800 ミリ秒
2. 1200 ミリ秒
3. 1500 ミリ秒
4. 1800 ミリ秒

```

void    taskA( INT stacd, VP exinf )
{
    for( ;; ){
        tk_slp_tsk( TMO_FEVR );
        /* 略：完了までに 100 ミリ秒かかる処理 */
    }
}

void    taskB( INT stacd, VP exinf )
{
    for( ;; ){
        tk_slp_tsk( TMO_FEVR );
        /* 略：完了までに 200 ミリ秒かかる処理 */
    }
}

void    taskC( INT stacd, VP exinf )
{
    for( ;; ){
        tk_slp_tsk( TMO_FEVR );
        /* 略：完了までに 400 ミリ秒かかる処理 */
    }
}

void    taskS( INT stacd, VP exinf )
{
    T_CTSK   ctskA = { NULL, TA_HLNG|TA_RNGO, (FP)taskA, 3, 1024, /*他のメンバは省略*/ };
    T_CTSK   ctskB = { NULL, TA_HLNG|TA_RNGO, (FP)taskB, 5, 1024, /*他のメンバは省略*/ };
    T_CTSK   ctskC = { NULL, TA_HLNG|TA_RNGO, (FP)taskC, 6, 1024, /*他のメンバは省略*/ };
    ID       tskidA, tskidB, tskidC;
    INT      i;

    tskidA = tk_cre_tsk(&ctskA);
    tskidB = tk_cre_tsk(&ctskB);
    tskidC = tk_cre_tsk(&ctskC);
    tk_sta_tsk(tskidA, 0);
    tk_sta_tsk(tskidB, 0);
    tk_sta_tsk(tskidC, 0);

    /* [ ア ] */

    for( i = 0; ; i++ ){
        if( (i % 6) == 0 ){
            tk_wup_tsk( tskidA );
        }
        if( ((i-1) % 8) == 0 ){
            tk_wup_tsk( tskidB );
        }
        if( ((i-2) % 6) == 0 ){
            tk_wup_tsk( tskidC );
        }
        tk_dly_tsk( 100 );
    }
    tk_ext_tsk();
}

```

(計算用紙)

## 問 2 5

T-Kernel を利用して書かれた次ページのプログラムを動作させるものとする。

このとき、下記の設問に解答せよ。

ただし、プログラムは以下の条件で動作するものとする。

- ・ 起動時には task\_init が優先度 1 のタスクとして実行される。
- ・ task\_init のシステムコールはすべて正常終了する。
- ・ task\_init が終了し、他のタスクの実行が開始された時刻を時刻 X とする。
- ・ 受信したメッセージの処理など、処理が省略されている部分ではシステムコールは発行されていないものとする。
- ・ T\_CMBX 構造体のメンバは次の通りとし、これ以外のメンバの記載は省略している。

```
typedef struct t_ctsk {
    VP    exinf;        /* 拡張情報 */
    ATR   mbxatr;       /* メールボックス属性 */
    /* これ以下のメンバの記載は省略 */
} T_CMBX;
```

- ・ T\_CTSK 構造体のメンバは次の通りとし、これ以外のメンバの記載は省略している。

```
typedef struct t_ctsk {
    VP    exinf;        /* 拡張情報 */
    ATR   tskatr;       /* タスク属性 */
    FP    task;         /* タスク起動アドレス */
    PRI   itskpri;      /* タスク起動時優先度 */
    /* これ以下のメンバの記載は省略 */
} T_CTSK;
```

※T-Kernel と  $\mu$ ITRON の仕様の相違点については、以下を前提として解答せよ。

- ・ T-Kernel の tk\_cre\_mbx は  $\mu$ ITRON の acre\_mbx に相当する。
- ・ T-Kernel の tk\_cre\_tsk は  $\mu$ ITRON の acre\_tsk に相当する。
- ・ T-Kernel の tk\_rcv\_mbx は  $\mu$ ITRON の trcv\_mbx に相当する。
- ・ サービスコールの名称は tk\_ext\_tsk を ext\_tsk のように読み替える。

### 【設問】

taskA が 2 回目に発行したメッセージの処理がタスク C、または、タスク D において完了するのは時刻 X から [ ア ] [ イ ] 0 ミリ秒後である。

[ ア ] [ イ ] に当てはまる数値をそれぞれ解答欄にマークしなさい。

ただし、その数が 3 桁ではなく 2 桁となる場合、[ ア ] は 0 をマークすること。

```

ID    mbxid;
ID    tskidA, tskidB, tskidC, tskidD;

void  task_init(INT stacd, VP exinf)
{
    T_CMBX  cmbx    = { NULL, TA_TFIFO|TA_MFIFO, /* 他のメンバは省略 */ };
    T_CTSK  ctskA   = { NULL, TA_HLNG, taskA, 2, /* 他のメンバは省略 */ };
    T_CTSK  ctskB   = { NULL, TA_HLNG, taskB, 3, /* 他のメンバは省略 */ };
    T_CTSK  ctskC   = { NULL, TA_HLNG, taskC, 4, /* 他のメンバは省略 */ };
    T_CTSK  ctskD   = { NULL, TA_HLNG, taskD, 5, /* 他のメンバは省略 */ };

    mbxid = tk_cre_mbx(&cmbx);

    tskidA = tk_cre_tsk(&ctskA);
    tskidB = tk_cre_tsk(&ctskB);
    tskidC = tk_cre_tsk(&ctskC);
    tskidD = tk_cre_tsk(&ctskD);
    tk_sta_tsk(tskidA, 0);
    tk_sta_tsk(tskidB, 0);
    tk_sta_tsk(tskidC, 0);
    tk_sta_tsk(tskidD, 0);

    tk_ext_tsk();
}

void  taskA(INT stacd, VP exinf)
{
    T_MSG   *pk_msg;

    while (1) {
        tk_dly_tsk(30);
        /* 略 : pk_msg の設定 */
        tk_snd_mbx(mbxid, pk_msg);
    }
}

void  taskB(INT stacd, VP exinf)
{
    T_MSG   *pk_msg;

    while (1) {
        tk_dly_tsk(50);
        /* 略 : pk_msg の設定 */
        tk_snd_mbx(mbxid, pk_msg);
    }
}

```

```

void taskC(INT stacd, VP exinf)
{
    T_MSG *pk_msg;
    ER ercd;

    while (1) {
        ercd = tk_rcv_mbx(mbxid, &pk_msg, 20);
        /* 略：受信したメッセージの処理（所要時間 20ms）*/
        tk_dly_tsk(30);
    }
}

void taskD(INT stacd, VP exinf)
{
    T_MSG *pk_msg;
    ER ercd;

    while (1) {
        ercd = tk_rcv_mbx(mbxid, &pk_msg, TMO_FEVR);
        /* 略：受信したメッセージの処理（所要時間 20ms）*/
    }
}

```

(計算用紙)



A. 解答上の注意

1. 氏名欄

氏名欄には、氏名、フリガナを記入しなさい。

2. 試験場欄

試験場欄は、受験している試験場所に○が付けられていることを確認しなさい。

(記入例) 東京で受験する場合

東京・大阪・他
---------

3. 受験日欄

受験日欄には、今日の年月日が記入されていることを確認しなさい。

(記入例) 2011年9月3日の場合

20	11	年	09	月	03	日
----	----	---	----	---	----	---

4. 受験番号欄

解答用紙の受験番号欄に受験番号が正しく記入され、さらにその下のマーク欄に正しくマークされていることを確認しなさい。

**正しくマークされていない場合は、採点できないことがあります。**

5. BまたはHBの黒鉛筆または黒色シャープペンシルを使用して正しく記入、マークしなさい。訂正の場合は、あとが残らないように消しゴムできれいに消し、消しくずを残さないようにしなさい。

6. 解答用紙は光学式読取装置で自動処理しますので、解答用紙のマーク例にしたがって正しくマークしなさい。

7. 問題文において「選択肢から一つ選べ」もしくは「一つずつ選べ」と指示されている場合は、解答欄に1つだけマークしなさい。

(記入例) 3を選択する場合

①	②	●	④
---	---	---	---

8. 問題文において「選択肢からすべて選べ」と指示されている場合は、解答欄に1つ以上マークしなさい。

(記入例) 2と4を選択する場合

①	●	③	●
---	---	---	---

9. 問題文において数値を選択することが指示されている場合は、解答欄では1つの桁に1つだけマークしなさい。

(記入例) 6を選択する場合

①	②	③	④	⑤	●	⑦	⑧	⑨	⑩
---	---	---	---	---	---	---	---	---	---

表紙に戻る場合は、この問題冊子を**右側から**裏返しなさい。

**試験開始の合図があるまで、この問題冊子を開いてはいけません。**